5

UNIT

XML

Compiled by:
LB Gurung

lbgurung00@gmail.com

❑ XML (eXtensible Markup Language)

lbgurung00@gmail.com

Introduction:

- ✓ XML stands for eXtensible Markup Language derived from SGML(Standard Generalized Markup Language)
- ✓ It is like HTML and was designed to store, describe and transport data
- ✓ XML tags are not predefined. You must define your own tags.
- ✓ XML uses a Document Type Definition(DTD) or schema to describe the data
- ✓ XML with a DTD or XML schema is designed to be self-descriptive
- ✓ XML is a cross platform, software and hardware independent tool for transmitting information

lbgurung00@gmail.com

❑ XML (eXtensible Markup Language)

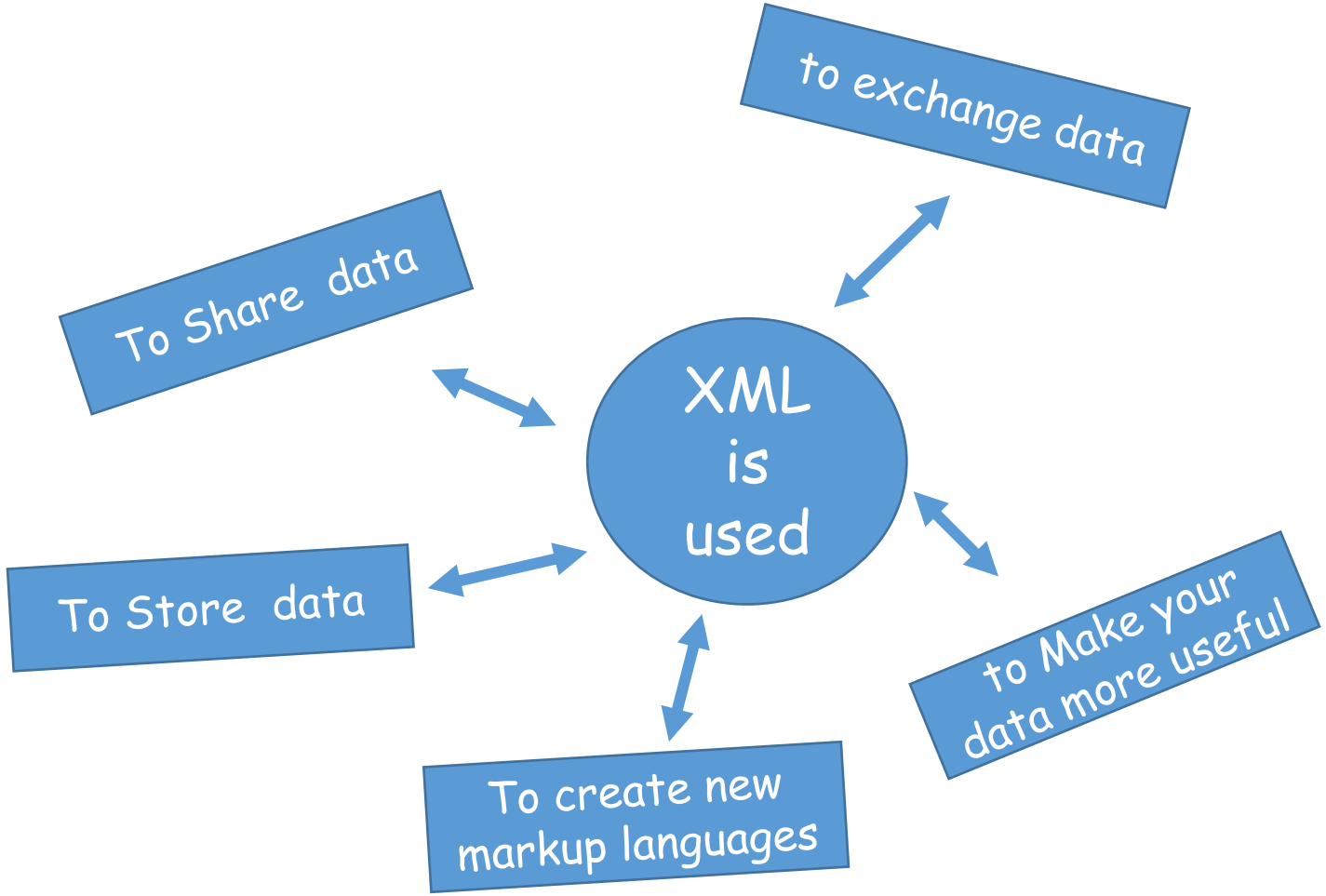
Introduction:

- ✓ It is an open-standards-based technology which is both human and machine readable. XML are best suited for use in documents that have large amount of similarity.
- ✓ The first version of XML(version 1.0) was announced by W3C in 1998. Version 1.1 came out in early 2004.

lbgurung00@gmail.com

- Where XML is used?

lbgurung00@gmail.com



❑ XML usage:

- ✓ XML can work behind the scene to simplify the creation of HTML documents for large web sites.
- ✓ XML can be used to exchange the information between organizations and systems.
- ✓ XML can be used for offloading and reloading of databases.
- ✓ XML can be used to store and arrange the data, which can customize your data handling needs.
- ✓ XML can easily be merged with style sheets to create almost any desired output.

lbgurung00@gmail.com

- Difference between HTML and XML

lbgurung00@gmail.com

HTML	XML
1. HTML is a predefined markup language and has a limited capability.	1. XML is a text-based markup language which has the self-describing structure and can effectively define another markup language
2. HTML was designed to display data and to focus on how data looks	2. XML was designed to describe data and to focus on what the data is
3. HTML is case insensitive	3. XML is case sensitive
4. HTML is used for designing a web-page to be rendered on the client side	4. XML is used to transport data between the application and the database
5. HTML is about displaying data hence static	5. XML is about carrying information , hence dynamic
6. Extension is .htm or .html	6. Extension is .xml
7. Closing tags are optional	7. Must have closing tag

❑ Advantages and Disadvantages of XML

Advantages:

- ✓ XML keeps content separate from presentation
- ✓ XML is an open format that can be read by many applications
- ✓ XML can be used to read on both the Client and the Server
- ✓ XML has widespread support in multiple languages , runtimes
- ✓ XML makes it possible for disparate systems to exchange data
- ✓ It supports Unicode, allowing almost any information in any written human language to be communicated.
- ✓ XML simplifies data sharing, data transport.

❑ Advantages and Disadvantages of XML

Disadvantages:

- ✓ XML is not suitable for very large data sets
- ✓ Storage of binary data such as image data is also inefficient
- ✓ XML doesn't support array
- ✓ XML file sizes are usually very large due to its verbose nature, it is totally dependent on who is writing it.

lbgurung00@gmail.com

❑ XML editors:

- ✓ XML is Case Sensitive so using **Normal Text Editor will create unnecessary Complications**.
- ✓ XML editors are similar to HTML Editor. XML Editors used to **highlight user defined tags which improves readability of XML** document. Process of writing XML document is somewhat simpler using XML Editor.
- ✓ **Different XML Editors :**
  - XML Notepad
  - XML Cooktop
  - XML Pro
  - XML Spy
  - Liquid XML Studio

lbgurung00@gmail.com

❑ XML Syntax:

```
<?xml version="1.0"?>
  <contact>
  <name>pop</name>
  <company>ABC</company>
 <phone>1234456</phone>
</contact>
```

lbgurung00@gmail.com

❑ Structure of XML Document:

lbgurung00@gmail.com

❑ XML Elements:

✓ XML elements contains of :

- Other element
- Text
- Attributes
- Mixing above elements

✓ XML documents must contain a **root element. This element is the parent of all other elements.**

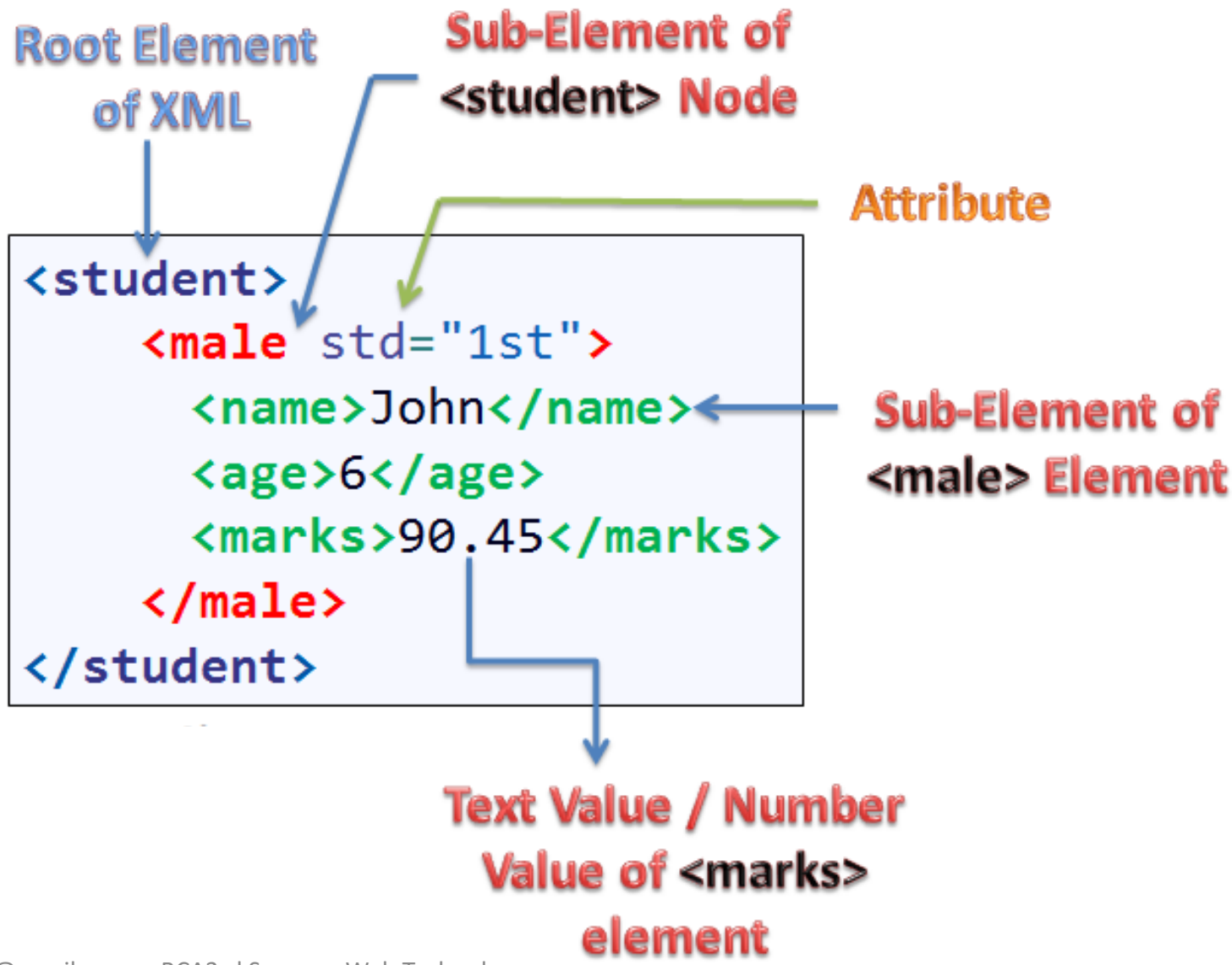
✓ The elements in an XML document form a document tree. The tree starts at the root and branches to the lowest level of the tree

✓ All elements can have sub-elements(child elements)

✓ XML elements are represented by tags

- Structure of XML Document:

For example:



- Rules for creating good XML

Basic rules for building good XML are:

- All XML must have a root element.

For example:

```
<book>
    <author>Smith </author>
</book>
```

- All tags must be closed

for example:

```
<author>John</author>
```

Rules for creating good XML

- All tags must be properly nested

For example:

<B><I>BCA</I></B> or

BCA

- Tag Names have strict limits: Tag names can't start with the letters *xml*, a number, or punctuation, except for the underscore character (_). For example:

<author> <01_author> wrong

<_author>

lbgurung00@gmail.com

Rules for creating good XML

- Tags names are case sensitive

For example:

<faculty>BCA</faculty>

<Faculty>BCA</Faculty>

- Tags name can't contain spaces: Spaces in tag names can cause all sorts of problems with data-intensive applications, so they're prohibited in XML.

lbgurung00@gmail.com

- Rules for creating good XML

lbgurung00@gmail.com

- Attributes value must appear within quotes

For example:

<note date="23/6/2020">

<chapter number="1">

<artist title="author" nationality="Nepal">

- White space is preserved: XML does not truncate multiple white-spaces

For example:

Hello Pop(in XML)

Hello Pop(in HTML)

- Rules for creating good XML

lbgurung00@gmail.com

- Avoid HTML Tags(optional): Because you can name tags anything you want, you could use tags reserved for HTML markup, such as <h1>, <p>, <li>, and so on. Although permissible in XML, avoid using such tag names unless you want the data to be formatted that way when it's viewed in a browser window.

Note: XML that follows these above rules is said to be "well formed XML Document". It means means that all the tags match up

XML Attributes

- Attributes can be added to XML elements to provide more information about the element.
- Attributes must have quoted values
- XML attributes specified by name="value" pair inside the starting element.
- For example:

```
<tutorials type="Web">
  <tutorial>
    <name>XML</name>
  </tutorial>
</tutorials>
```

❑ XML comments

- **XML comments** are similar to HTML comments.
- A comment is a character, a line or a paragraph that is not considered as part of the XML code that needs to be processed.
- A comment allows you to insert notes or personal observations inside an XML file.
- The comments are added as notes or lines for understanding the purpose of an XML code.
- They are visible only in the source code; not in the XML code. Comments may appear anywhere in XML code.

❑ XML comments

- **Syntax:**

<! --your comment -->

Example:

xml prolog

```
<?xml version="1.0" encoding="UTF-8"?>
<!--students grades-->
<class>
<student>
<name>Khem</name>
<grade>A+</grade>
</student>
</class>
```

- XML comments rules:

Following rules should be followed for XML comments –

- ✓ Comments cannot appear before XML declaration.
- ✓ Comments may appear anywhere in a document.
- ✓ Comments must not appear within attribute values.
- ✓ Comments cannot be nested inside the other comments.

Note: **UTF** stands for Unicode Transformation Format. The '**8**' means it **uses 8**-bit blocks to represent a character.

- What is a well-formed XML document?

If a document is syntactically correct it can be called as well-formed XML documents. A well-formed document conforms to XML's basic rules of syntax:

- ✓ Every open tag must be closed.
- ✓ The open tag must exactly match the closing tag:XML is case-sensitive.
- ✓ All elements must be embedded within a single root element.
- ✓ Child tags must be closed before parent tags.
- ✓ A well-formed document has correct XML tag syntax, but the elements might be invalid for the specified document type.

XML Elements vs Attributes

Example:

```
<person sex="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

```
<person>
  <sex>female</sex>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

- In the first example sex is an attribute. In the last, sex is an element. Both examples provide the same information.

❑ XML Namespaces

- ✓ It is a special type of reserved XML attribute that you place in an XML tag.
- ✓ It is a set of unique names which is used to avoid element name conflict in XML document.
- ✓ Identified by URI(Uniform Resource Identifier)
- ✓ Attribute name must start with " xmlns: " which stands for XML namespace.
- ✓ Syntax: <element xmlns:name="URL"

  here name is prefix
- ✓ In XML, element names are defined by the developer, it may results in a conflict when we mix XML documents from different XML applications

❑ XML Namespaces

✓ Example of conflict:

1.xml

```
<class>
    <name>-------</name>
</class>
```

2.Xml

```
<class>
    <name>------- </name>
</class>
```

Here, a conflict occurs due to same element name

Now, XML Namespace is used to avoid the conflict

❑ XML Namespaces

✓ Example of namespace:

1.xml

```
<?xml version="1.0">
<c1.class xmlns:c1="http://www.abc.com/url1">
    <c1.name>Peter</c1.name>
</c1.class>
```

prefix

2.xml

```
<?xml version="1.0">
 <c2.class xmlns:c2="http://www.xyz.com/url2">
   <c2.name>Peter</c2.name>
 </c2.class>
```

Now ,there will be no conflict due to namespace

❑ XML Namespaces

✓ Example of namespace:

```
<?xml version="1.0"?>
<engineering>
    <subject> OS </subject>
</engineering>
```

```
<?xml version="1.0"?>
<medical>
    <subject> Biochemistry </subject>
</medical>
```

If these XML fragments were added together , then would be a name conflict

Both contain a <subject> element, but the elements have different content and meaning

To solve this name conflicts, XML namespace is used

❑ XML Namespaces

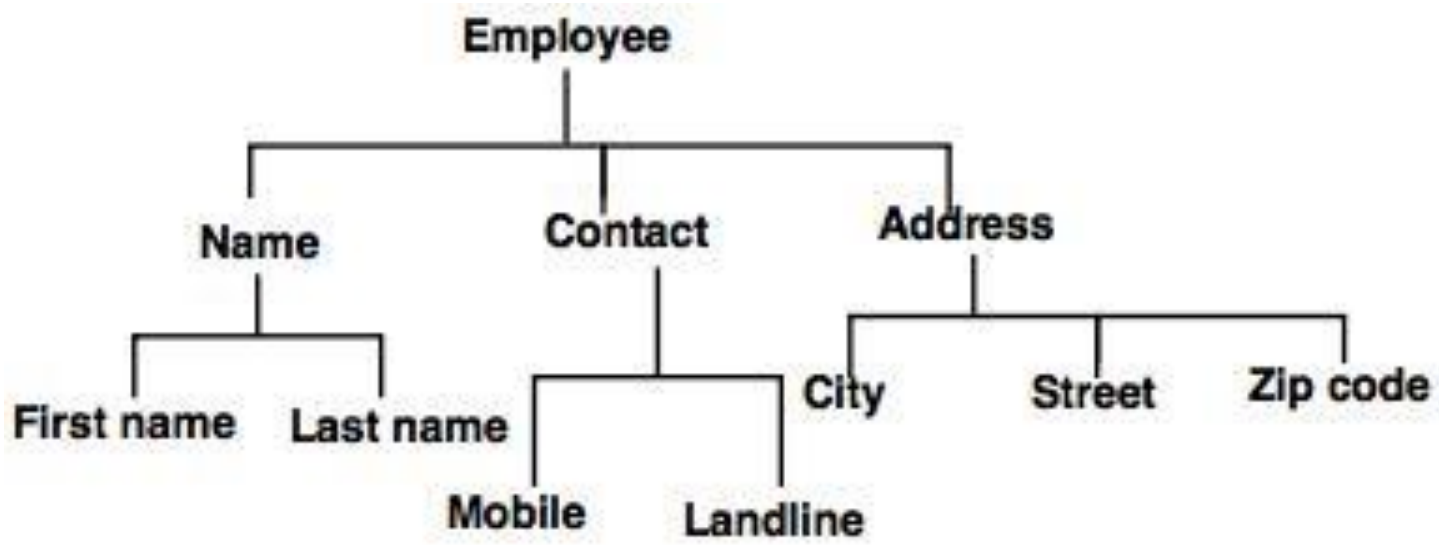
```
<?xml version="1.0"?>
<college>
   <eng:engineering xmlns:eng="http://www.abc/a1">
         <eng:subject>OS</eng:subject>
   </eng:engineering>

<med:medical xmlns:med= "http://www.xyz/b1">
  <med:subject>Biochemistry</med:subject>
</med:medical>
</college>
```

❑ XML Tree structure

- ✓ An XML document has a self descriptive structure. It forms a tree structure which is referred as an XML tree.
- ✓ XML tree structure is also called as tree model or hierarchical model.
- ✓ XML document must contain a root element and all elements in the document can contain sub-elements, text and attributes.
- ✓ The complex elements are represented with the help of internal nodes and simple elements are represented with leaf node.

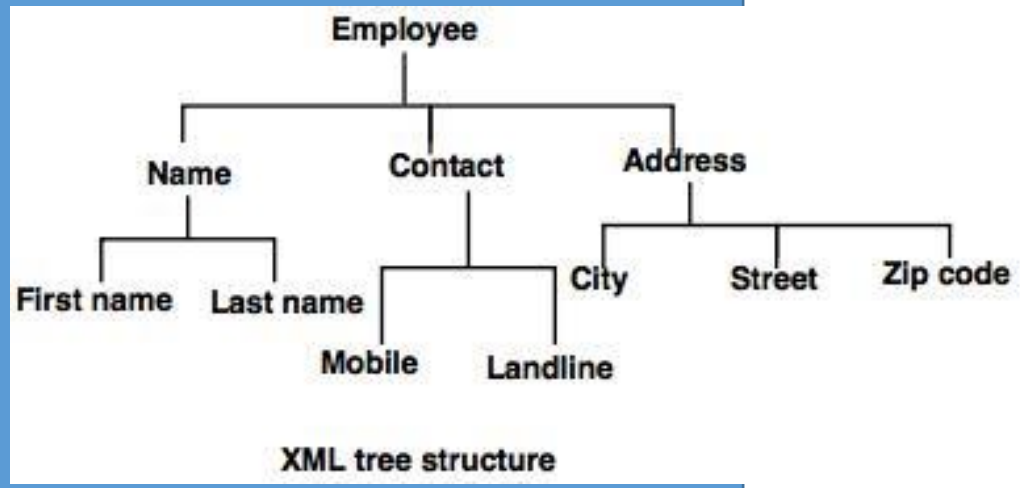
❑ XML Tree structure

✓ **Example:** In the following example, employee information is represented with tree structure, which can be used in XML document.



XML tree structure

❑ XML Tree structure



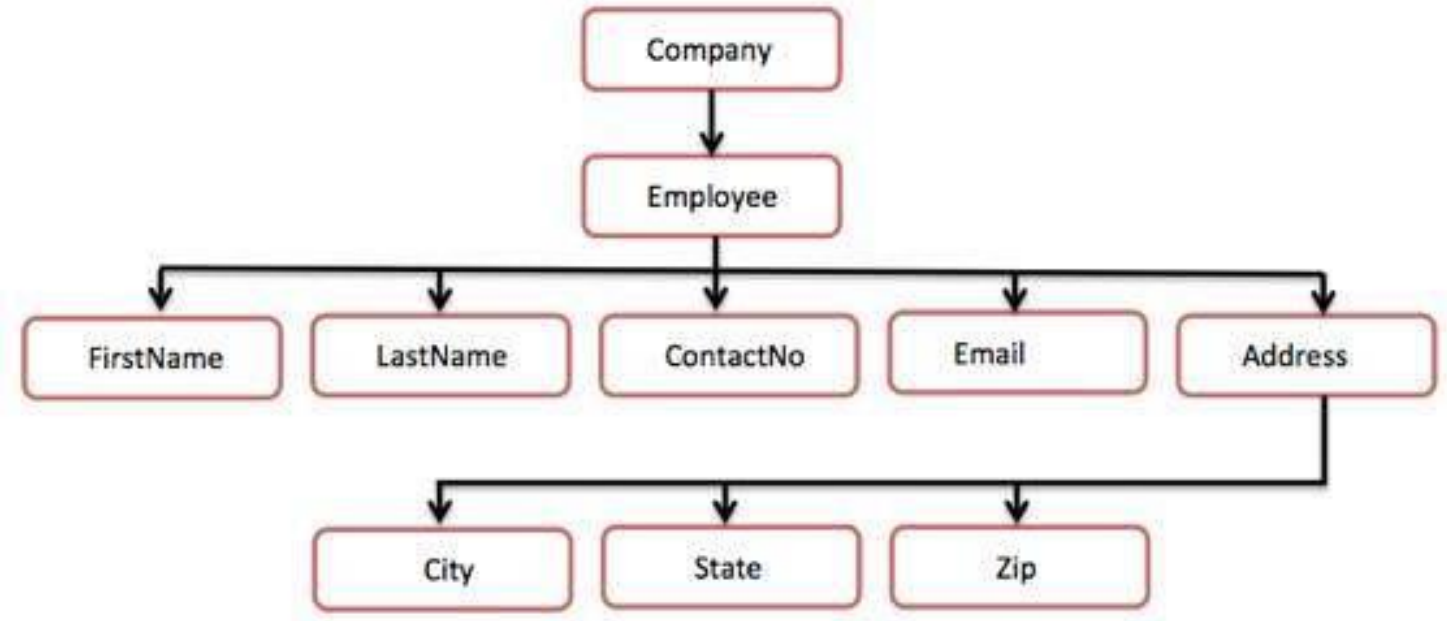
XML tree structure

✓ Textual representation of the above document in the form of XML document can be as follows:

```
<?xml version="1.0"?>
<Employee>
    <Name>
        <Firstname>Bob</Firstname>
        <Lastname>Shrestha<</Lastname>
    </Name>
    <Contact>
        <Mobile>9842011111</Mobile>
        <Landline>02152332</Landline>
    </Contact>
    <Address>
        <City>Brt</City>
        <Street>colz road</Street>
        <Zip code>00977</Zip code>
    </Address>
</Employee>
```

- XML Tree structure

Write well-formed XML document for the following structure



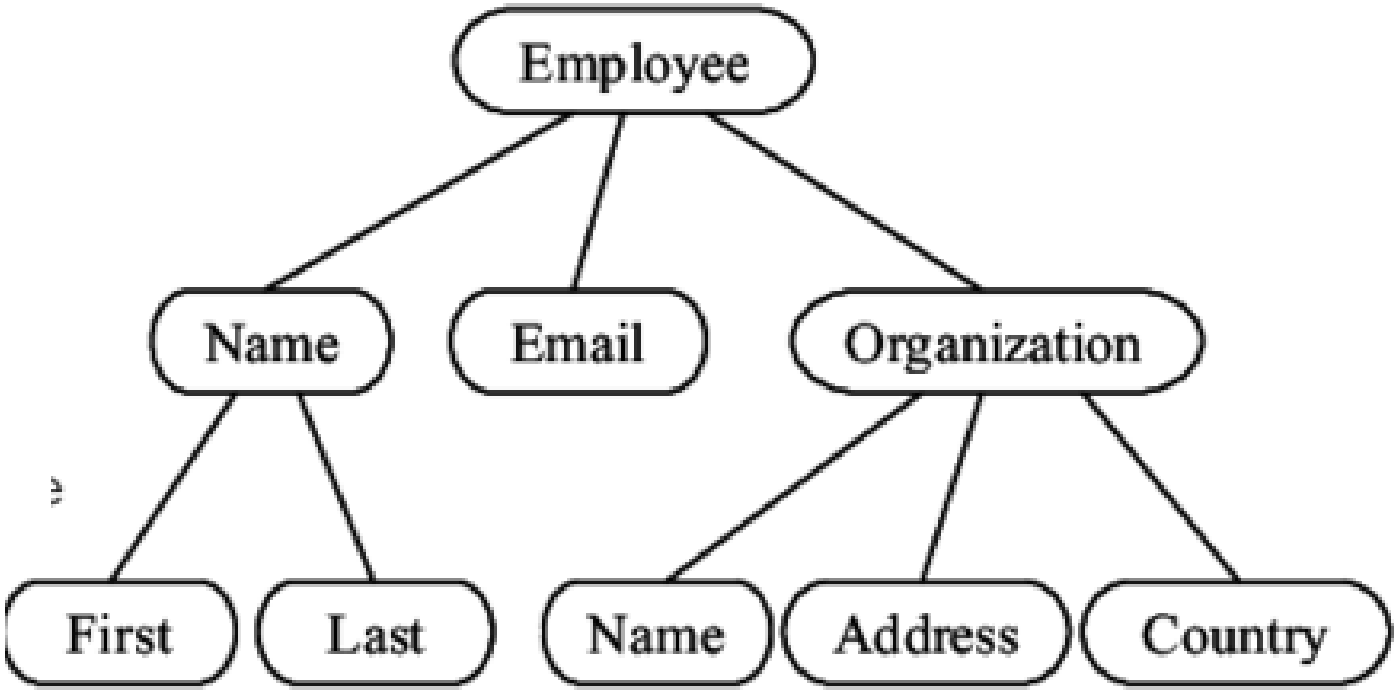
- XML Tree structure

Write well-formed XML document for the following structure

```
<?xml version = "1.0"?>
<Company>
<Employee>
<FirstName>Pop</FirstName>
<LastName>Rai</LastName>
<ContactNo>1234567890</ContactNo>
<Email>pop@gmail.com</Email>
<Address>
<City>Bangalore</City>
<State>Karnataka</State>
<Zip>560212</Zip>
 </Address>
</Employee>
</Company>
```

- XML Tree structure

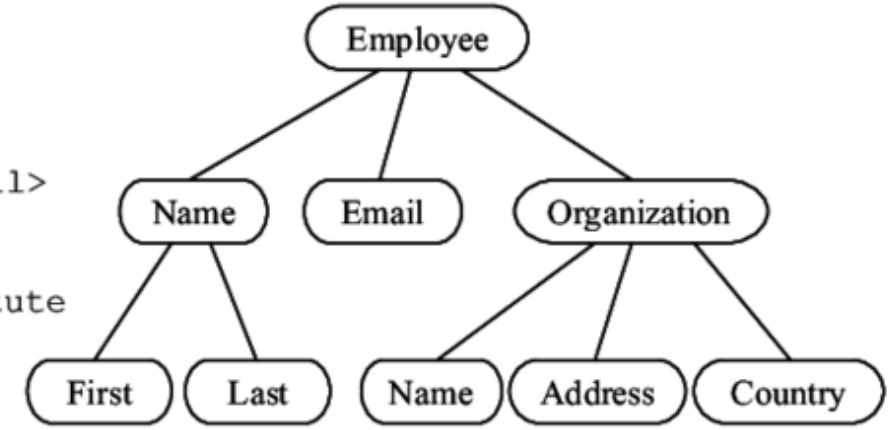
Write well-formed XML document for the following structure



- XML Tree structure

Write well-formed XML document for the following structure

```
<Employee>
    <Name>
        <First>Lassi</First>
        <Last>Lehto</Last>
    </Name>
    <Email>Lassi.Lehto@fgi.fi</Email>
    <Organization>
        <Name>
            Finnish Geodetic Institute
        </Name>
        <Address>
            PO Box 15,
            FIN-02431 Masala
        </Address>
        <Country CountryCode="358">Finland</Country>
    </Organization>
</Employee>
```



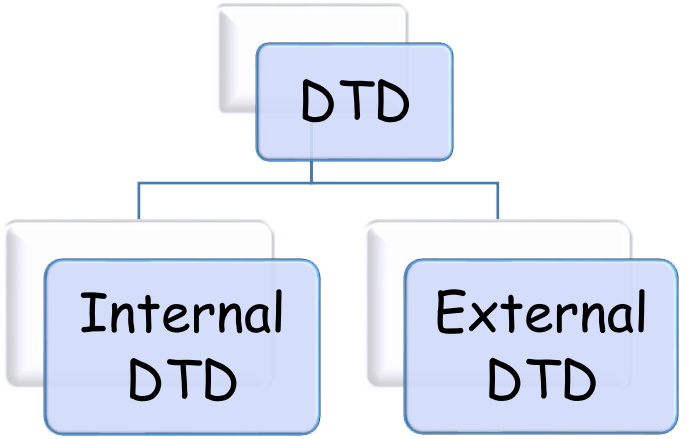
- XML DTD(Document Type Definition/Declaration)

- ✓ The XML Document Type Definition or Declaration, commonly known as DTD, is a way to describe XML language precisely (accurately/exactly).
- ✓ It is used define the structure of XML document
- ✓ It defines the document structure with the list of legal element and attributes of an XML document
- ✓ Used to perform validation

Syntax:

```
<!DOCTYPE element DTD identifier
[
declaration1
declaration2
.....
]>
```

❑ Types of DTD:



- Internal DTD:

✓ In Internal DTD , XML and DTD are in the same file.

✓ Elements are declared within the XML files

✓ Useful for simple documents

❑ Types of DTD:

Syntax:

<!DOCTYPE root-element

[element-declaration] >

There are two data types, PCDATA and CDATA

PCDATA: Parsed Character Data

CDATA: Character data, Not usually parsed

❑ Types of DTD:

- Internal DTD:

✓ Example:

```
<?xml version="1.0" encoding ="UTF-8" ?>
<!DOCTYPE Address [
<!ELEMENT Address(Name, Company, Phone)>
<!ELEMENT Name(#PCDATA)>
<!ELEMENT Company(#PCDATA)>
<!ELEMENT Phone(#PCDATA)>
]>
<Address>
        <Name>Peter</Name>
        <Company>xyz</Company>
        <Phone>123456</Phone>
</Address>
```

DTD

XML

❑ Types of DTD:

- External DTD:

✓ A DTD that resides in a file (.dtd extension) other than the XML file

✓ Elements are declared outside the XML files

✓ External DTD are reusable

✓ More common for professional documents

Syntax:

<!DOCTYPE root-element SYSTEM "file-name">

❑ Types of DTD:

- External DTD:

✓ Example:

Add.dtd

```
<!DOCTYPE Address [
<!ELEMENT Address(Name, Company, Phone)>
<!ELEMENT Name(#PCDATA)>
<!ELEMENT Company(#PCDATA)>
<!ELEMENT Phone(#PCDATA)>
]>
```

XML file:

```
<?xml version="1.0" encoding UTF-8?>
<!DOCTYPE Address SYSTEM "Add.dtd">
<Address>
<Name>Peter</Name>
<Company>xyz</Company>
<Phone>123456</Phone>
</Address>
```

❑ Types of DTD:

- Internal DTD:

✓ Example:

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<!DOCTYPE Airline [
<!ELEMENT Airline (aname, country, class)>
<!ELEMENT aname (#PCDATA)>
<!ELEMENT country (#PCDATA)>
<!ELEMENT class (#PCDATA)>
]>
        <Airline>
        <aname>Japan</aname>
        <country>Italy</country>
        <class>passengers-business</class>
        </Airline>
```

❑ Use of DTD:

- Independent groups of people can agree to use a standard DTD for interchanging data
- We can verify that the received data from other person is valid or not
- We can also verify our own data

❑ Write an XML code to accept student details [Name, ID, Branch and CGPA]

```
<?xml version="1.0" encoding="UTF-8"?>
<class>
 <student>
<name> ABC </name>
 <id> 001 </id>
<branch> IT </branch>
<cgpa> 9 </cgpa>
</student>
</class>
```

❑ For the following XML document, create the DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<Addressbook>
 <Contact>
<name> Bob </name>
 <Address> Brt-1 </Address>
<City> Brt </City>
<Pin> 0977 </Pin>
<Phone>9852011111</Phone>
</Contact>
</Addressbook>
```

DTD file

```
<!ELEMENT Addressbook (Contact)>
<!ELEMENT Contact(Name, Address ,City,Pin, Phone)>
<!ELEMENT Name(#PCDATA)>
<!ELEMENT Address(#PCDATA)>
<!ELEMENT City(#PCDATA)>
<!ELEMENT Pin(#PCDATA)>
<!ELEMENT Phone(#PCDATA)>
```

- **xml file**

```
<?xml version="1.0” encoding=“UTF-8”?>
<Furniture>
<Sofas>soft</Sofas>
<Chairs>medium height</Chairs>
<Table>Accounting</Table>
</Furniture>
```

- **DTD file**

```
<!ELEMENT Furniture (Sofas, Chairs, Table)>
<!ELEMENT Sofas (#PCDATA)>
<!ELEMENT Chairs (#PCDATA)>
<!ELEMENT Table (#PCDATA)>
]>
```

❑ XML Schema:

- XML Schema is commonly known as XML Schema Definition(XSD).
- It is used to describe and validate the structure and the content of XML data.
- XML schema defines the elements , attributes and data types .
- The purpose of an XML schema is to define the legal building blocks of an XML document, just like a DTD.
- An XML schema is used to define the structure of an XML document. It is like DTD but provides more control on XML structure.
- An XML document with correct syntax is called "Well Formed".
- An XML document validated against an XML Schema is both "Well Formed" and "Valid".

❑ XSD Element

- XML Schema defines the element of XML files.

There are two types of elements:

a. Simple element

b. Complex element

- **XSD simple element:**

✓ A simple element is an XML element that contain only text

✓ It can't contain any other elements or attribute

✓ XML schema has a lot of built-in data types like:

xs:string , xs:decimal , xs:integer , xs:Boolean, xs:date, xs:time

Example:

❑ XSD Element

XML Elements

<name>Peter</name>

<age>23</age>

<mobile>122345</mobile>

Simple element definitions

<xs:element name="fname" type="xs:string"/>

<xs:element name="age" type="xs:integer"/>

<xs:element name="mobile" type="xs:integer"/>

❑ XSD Element

Simple type:

xml :

<name>Peter</name>
<age>21</age>
<dob>2020-07-08</dob>

DTD for the above:

<!ELEMENT name(#PCDATA)>
<!ELEMENT age(#PCDATA)>
<!ELEMENT dob(#PCDATA)>

XSD for the above:

<xs:element name="name" type="xs:string" />
<xs:element name="age" type="xs:integer" />
<xs:element name="dob" type="xs:date" />

❑ XSD Element

- **XSD Complex element:**

✓ A complex element is an XML element that contain other elements and / or attributes.

✓ Types of XSD Complex element:

There are four types of element:

a. Empty element:

An empty element is an element that can have only attribute, but no content.

For example:

<product pid="123" /> ,

<student rollno="101" />

- XSD Element

a. Empty element:

For example:

<product pid="123" /> ,

<student rollno="101" />

The schema for the above example:

```
<xs:element name="product">
    <xs:complextype>
        <xs:attribute name="pid" type="xs:positiveinteger" />
  </xs:complextype>
</xs:element>
```

❑ XSD Element

b. Elements that contain only other elements:

For example:

```
<person>
    <name>Peter</name>
    <age> 21</age>
</person>
```

Schema for the above:

```
<xs:element name="person">
   <xs:complextype>
       <xs:sequence>
          <xs:element name="name" type="xs:string" />
          <xs:element name="age" type="xs:integer" />
    </xs:sequence>
 </xs:complextype>
</xs:element>
```

- XSD Element

c. Elements that contain only text:

For example:

<food type="dessert">Ice-cream</food>

<shoesize country="Nepal">41</shoesize>

- XSD Element

c. Elements that contain only text:

For example:

```
<xs:element name="shoesize">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:integer">
        <xs:attribute name="country" type="xs:string" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

❑ XSD Element

```
<xs:element name="letter">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="orderid" type="xs:positiveInteger"/>
      <xs:element name="shipdate" type="xs:date"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

❑ XSD Element

d. Elements that contain both other elements and text:

For example:

```
<letter>
  Dear Mr. <name>John </name>.
  Your order <orderid>101</orderid>
  will be shipped on <shipdate>2020-07-13</shipdate>.
</letter>
```

❑ XSD Attribute

- ✓ Simply attributes are associated with the complex element . If an element has attributes , it is considered to be of a Complex type
- ✓ Simple elements can not have attributes. But the attribute itself is always declared as a simple type. All attributes are declared as simple types.
- ✓ Syntax:

<xs:attribute name="xyz" type="yyy" />

Here, xyz is the name of the attribute and yyy specifies the data type of the attribute

xml element with an attribute:

<name lname="Rai">Peter</name>

Here the corresponding attribute definition:

<xs:attribute name="lname" type="xs:string" />

❑ XSD Attribute

✓ Default and Fixed Values for Attributes:

- Attributes may have a default value or a Fixed value specified
- A default value is automatically assigned to the attribute when no other value is specified
- In the following example, the default value is "Englsih".

```
<xs:attribute name="lang" type="xs:string" default="English" />
```

- A fixed value is automatically assigned to the attribute, and you cannot specify another value
- For example:

```
<xs:attribute name="lang" type="xs:string"  fixed="English" />
```

❑ XSD Attribute

✓ Restrictions on content:

- When XML element or attribute has a data type defined, it puts restrictions on the element's or attributes content. If an XML element is of type "date" and contains a string like "Hello", the element will not validate.
- With XML schemas , you can also add your own restrictions to your XML elements and attributes. These restrictions are called **facets** **or**
- Restrictions are used to define acceptable values for XML elements or attributes. Restrictions on XML elements are called facets.

❑ XSD Attribute

✓ Restrictions on content:

- When XML element or attribute has a data type defined, it puts restrictions on the element's or attributes content. If an XML element is of type "date" and contains a string like "Hello", the element will not validate.
- With XML schemas , you can also add your own restrictions to your XML elements and attributes. These restrictions are called **facets** **or**
- Restrictions are used to define acceptable values for XML elements or attributes. Restrictions on XML elements are called facets.

- XSD Attribute

XML Restrictions /Facets

a. Restrictions on Values

- The following example defines an element called "age" with a restriction. The value of age cannot be lower than 0 or greater than 120:

```
<xs:element name="age">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="120"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

❑ XSD Attribute

b. Restrictions on a Set of Values

- To limit the content of an XML element to a set of acceptable values, we would use the enumeration constraint.
- The example below defines an element called "car" with a restriction. The only acceptable values are: Audi, Golf, BMW:

```
<xs:element name="car">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Audi"/>
      <xs:enumeration value="Golf"/>
      <xs:enumeration value="BMW"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

❑ XSD Attribute

b. Restrictions on a Set of Values

- To limit the content of an XML element to a set of acceptable values, we would use the enumeration constraint.
- The example below defines an element called "car" with a restriction. The only acceptable values are: Audi, Golf, BMW:

```
<xs:element name="car">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Audi"/>
      <xs:enumeration value="Golf"/>
      <xs:enumeration value="BMW"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

❑ XSD Attribute

- The next example defines an element called "zipcode" with a restriction. The only acceptable value is FIVE digits in a sequence, and each digit must be in a range from 0 to 9:

```
<xs:element name="zipcode">
 <xs:simpleType>
  <xs:restriction base="xs:integer">
   <xs:pattern value="[0-9][0-9][0-9][0-9][0-9]"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

❑ XSD Attribute

c. Restrictions on a Series of Values

```
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="([a-z])*"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

The above example defines an element called "letter" with a restriction. The acceptable value is zero or more occurrences of lowercase letters from a to z:

- XSD Attribute

d. Restrictions on a Series of Values

```
<xs:element name="gender">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="male|female"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

The above example defines an element called "gender" with a restriction. The only acceptable value is male OR female

❑ XSD Attribute

e. Restrictions on Whitespace Characters

```
<xs:element name="address">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="preserve"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

The above example defines an element called "address" with a restriction. The whiteSpace constraint is set to "preserve", which means that the XML processor WILL NOT remove any white space characters:

❑ XSD Attribute

f. Restrictions on Length

✓ To limit the length of a value in an element, we would use the length, maxLength, and minLength constraints.

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

The above example defines an element called "password" with a restriction. The value must be exactly eight characters:

- XSD Attribute

f. Restrictions on Length

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="6"/>
      <xs:maxLength value="9"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

The above example defines another element called "password" with a restriction. The value must be minimum six characters and maximum nine characters:

❑ Write a XML code to store following information about Book

- Library has multiple books with single book
- Single book have ISBN attribute
- Book also have name,author, page and price element
- Books can be written by multiple author with author_name, email
- Price have currency attribute with NPR and usd choice with default NPR value

Xml file:

```
<?xml version="1.0"?>
<!DOCTYPE books SYSTEM "book1.dtd">
<books>
        <book isbn="1234">
            <name>OS</name>
            <author>
            <author_name>Peter</author_name>
            <email>peter@gmail.com</email>
            </author>
            <page>150</page>
            <price currency="NPR">500</price>
            </book>
    <book isbn="2345">
            <name>Web Tech.</name>
            <author>
            <author_name>James</author_name>
            <email>jam@gmail.com</email>
            </author>
<author>
<author_name>Pop</author_name>
<email>pp@gmail.com</email>
</author>
<page>300</page>
<price currency="NPR">1200</price>
</book>
```

```
<book isbn="1111">
        <name>HTML</name>
        <author>
        <author_name>Bob</author_name>
        <email>bob@gmail.com</email>
        </author>
        <page>250</page>
        <price currency="usd">50</price>
        </book>
</books>
```

DTD file:

```
<!ELEMENT books(book+)>

<!ELEMENT book(name,author+,page,price)>

<!ELEMENT name(#PCDATA)>

<!ELEMENT author(author_name,email)>

<!ELEMENT author_name(#PCDATA)>

<!ELEMENT email(#PCDATA)>

<!ELEMENT page(#PCDATA)>

<!ELEMENT price(#PCDATA)>

<!ATTLIST book isbn CDATA #REQUIRED>

<!ATTLIST price currency(NPR|usd) “NPR”>
```

2B.xml

C: > xampp > htdocs > bca > 2B.xml

```xml
<?xml version="1.0"?>
<!DOCTYPE books SYSTEM "book1.dtd">
<books>
        <book isbn="1234">
    <name>OS</name>
    <author>
    <author_name>Peter</author_name>
    <email>peter@gmail.com</email>
    </author>
    <page>150</page>
    <price currency="NPR">500</price>
    </book>
     <book isbn="2345">
    <name>Web Tech.</name>
    <author>
    <author_name>James</author_name>
    <email>jam@gmail.com</email>
    </author>
<author>
<author_name>Pop</author_name>
<email>pp@gmail.com</email>
</author>
<page>300</page>
<price currency="NPR">1200</price>
```

xml file:

2B.xml

C: > xampp > htdocs > bca > 2B.xml

```xml
<?xml version="1.0"?>
<!DOCTYPE books SYSTEM "book1.dtd">
<books>
        <book isbn="1234">
    <name>OS</name>
    <author>
    <author_name>Peter</author_name>
    <email>peter@gmail.com</email>
    </author>
    <page>150</page>
    <price currency="NPR">500</price>
    </book>
     <book isbn="2345">
    <name>Web Tech.</name>
    <author>
    <author_name>James</author_name>
    <email>jam@gmail.com</email>
    </author>
<author>
<author_name>Pop</author_name>
<email>pp@gmail.com</email>
</author>
<page>300</page>
<price currency="NPR">1200</price>
</book>
<book isbn="1111">
    <name>HTML</name>
    <author>
    <author_name>Bob</author_name>
    <email>bob@gmail.com</email>
    </author>
    <page>250</page>
    <price currency="usd">50</price>
    </book>
</books>

```

dtd file:

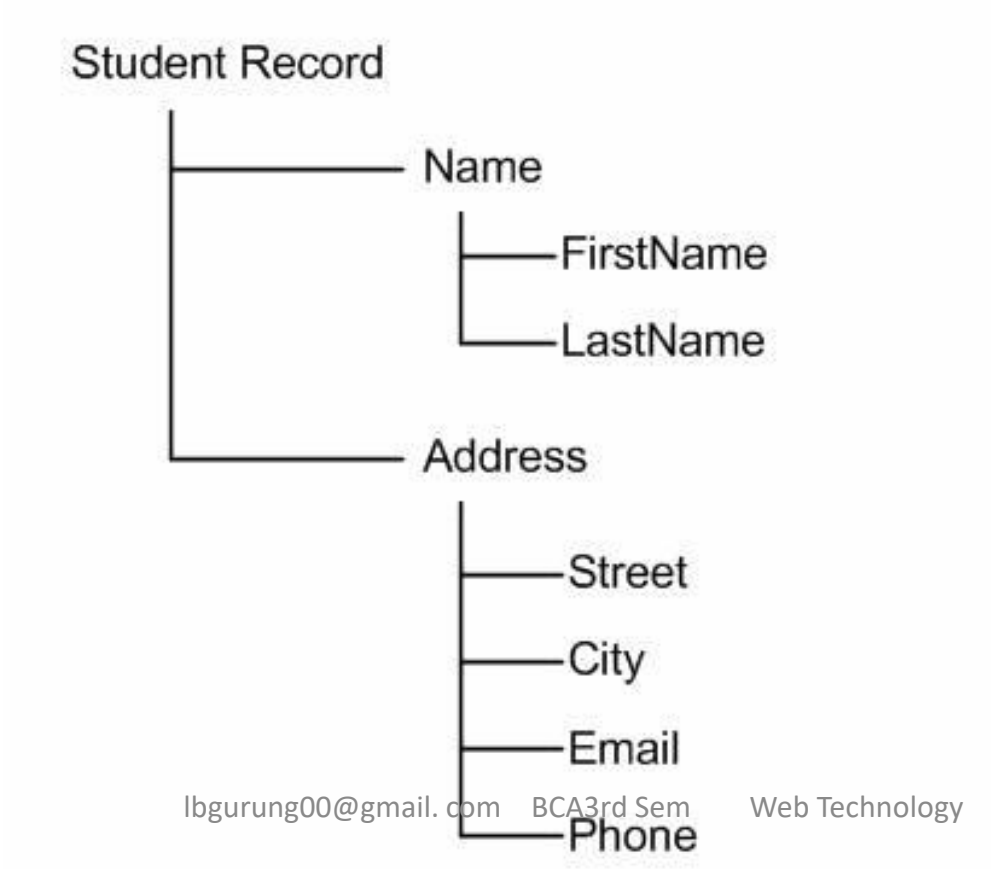
C: > xampp > htdocs > bca > book1.dtd

```
1    <!ELEMENT books(book+)>
2    <!ELEMENT book(name,author+,page,price)>
3    <!ELEMENT name(#PCDATA)>
4    <!ELEMENT author(author_name,email)>
5    <!ELEMENT author_name(#PCDATA)>
6    <!ELEMENT email(#PCDATA)>
7    <!ELEMENT page(#PCDATA)>
8    <!ELEMENT price(#PCDATA)>
9    <!ATTLIST book isbn CDATA #REQUIRED>
10   <!ATTLIST price currency(NPR|usd) "NPR">
11
```

Use XML to create a student record database for the student information management.

- Each student record includes student's name and address.
- The name has two parts: First name and last name.
- The address has four parts: Street, City, Email and Phone Number

The structure of the student record can be presented as:



```
<?xml version = "1.0"?>
<!--student1.xml-->
<students>
   <student>
    <name>
        <firstname> Peter </firstname>
        <lastname> Baral </lastname>
</name>
<address>
       <street> 101 South Street</street>
       <city> Brt </city>
        <email> pb@gmail.com </email>
         <phone> 253456 </phone>
</address>
</student>
<student>
     <name>
        <firstname> Tom </firstname>
         <lastname> Shrestha </lastname>
   </name>
<address>
     <street> 202 College Road </street>
      <city> Brt </city>
     <email> tom@gmail.com</email>
       <phone> 2213456 </phone>
</address>
</student>
</students>
```

DTD file

```
<?xml version = "1.0"?>
<!--students.dtd-a document type definition for the students.xml-->
<!ELEMENT students (student+)>
<!ELEMENT student (name,address)>
<!ELEMENT name (firstname,lastname)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT address (street,city,email,phone)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT Phone(#PCDATA)>
```

Note: To use the DTD file, we must add this code into the XML file.

```
<!DOCTYPE students SYSTEM "students.dtd">
```

Note: To use the DTD file, we must add this code into the XML file.

<!DOCTYPE students SYSTEM "students.dtd">

```
<?xml version = "1.0"?>
<!DOCTYPE students SYSTEM "students.dtd">
   <student>
    <name>
         <firstname> Peter </firstname>
         <lastname> Baral </lastname>
</name>
<address>
       <street> 101 South Street</street>
       <city> Brt </city>
        <email> pb@gmail.com </email>
         <phone> 253456 </phone>
</address>
</student>
<student>
      <name>
         <firstname> Tom </firstname>
          <lastname> Shrestha </lastname>
    </name>
<address>
      <street> 202 College Road </street>
        <city> Brt </city>
       <email> tom@gmail.com</email>
         <phone> 2213456 </phone>
</address>
</student>
</students>
```